~~LPS~~ and a probability estimate for the LPS, and a decoding engine that includes a range register to assign a value to a range for the LPS. The value is based on the probability estimate, a value stored in the range register and the context identifier to a range for the LPS if the context identifier is not equal to an index and the value is not based on the value stored in range register if the context identifier is equal to the index. The decoding engine further determines a value of a binary event based on the value of the range for the LPS and bits from an information sequence.

IN THE CLAIMS

This listing of claims will replace all prior versions, listings, of claims in the application.

1. (currently amended) An arithmetic decoder comprising:

a sequencer to generate a context identifier for an event of an event sequence; and

a probability estimator to determine a value for a least probable symbol (LPS) ~~LPS~~ and a probability estimate for the LPS; and

a decoding engine including a range register to assign a value to a range for the LPS, wherein the value is based on the probability estimate, a value stored in the range register and the context identifier to a range for the LPS if the context identifier is not equal to an index and the value is not based on the value stored in range register if the context identifier is equal to the index, and the decoding engine further to determine a value of a binary event based on the value of the range for the LPS and bits from an information sequence.

2. (original) The arithmetic decoder defined in Claim 1 wherein the decoding engine stops decoding when the context identifier is equal to the index and a LPS is decoded.

3. (original) The arithmetic decoder defined in Claim 2 wherein non-arithmetically encoded data follows the arithmetically coded data in the information sequence.

4. (original) The arithmetic decoder defined in Claim 2 wherein the index represents an end of slice indicator.

5. (original) The arithmetic decoder defined in Claim 1 wherein the decoding engine includes a value register and when the context identifier is equal to the index decodes the event based on the value in the value register by generating an event in a first state if the value in the value register is less than the number assigned to the LPS range or generating an event in a second state if the value in the value register is greater than or equal to the number.

6. (original) The arithmetic decoder defined in Claim 5 wherein the decoding engine performs renormalization in response to decoding the event only when the value in the value register is greater than or equal to the number.

7. (currently amended) The arithmetic decoder defined in Claim 1 wherein the decoding engine includes a value register and when the context ~~identifer~~ identifier is equal to the index decodes the event by first subtracting the value assigned to the LPS range from the range register and, wherein the event is generated in a first state if the value in the value register is greater than or equal to a value in the range register or is generated in a second state if the value in the value register is less than the value in the range register.

8. (original) The arithmetic decoder defined in Claim 7 wherein the decoding engine performs renormalization in response to decoding the event only when the value in the value register is greater than or equal to the value in the range register.

9. (original) The arithmetic decoder defined in Claim 7 wherein the value assigned to the range of the LPS is 2 if the context identifier is equal to the index.

10. (original) The arithmetic decoder defined in Claim 8 wherein the value assigned to the range of the LPS is 100 Hex if the context identifier is equal to the index.

11. (original) The arithmetic decoder defined in Claim 7 wherein the value assigned to the range of the LPS is 2 if the context identifier is equal to the index and wherein a last bit read in a bitstream containing binary events being decoded by the decoding engine is equal to 1.

12. (currently amended) A decoding method comprising:

generating a context identifier for a binary event of an event sequence;

determining a value for a least probable symbol (LPS) ~~LPS~~ and a probability estimate for the LPS;

assigning a value to a range for the LPS, wherein the value is based on the probability estimate, a value stored in the range register and the context identifier to a range for the LPS if the context identifier is not equal to an index and the value is not based on the value stored in range register if the context identifier is equal to the index; and

determining a value of a binary event based on the value of the range for the LPS and bits from an information sequence.

13. (original) The method defined in Claim 12 further comprising stopping decoding when the context identifier is equal to the index and a LPS is decoded.

14. (original) The method defined in Claim 12 wherein non-arithmetically coded data follows the arithmetically encoded data in the information sequence.

15. (original) The method defined in Claim 12 wherein the index represents an end of slice indicator.

16. (original) The method defined in Claim 12 wherein determining a value of a binary event based on the value of the range for the LPS and bits from an information sequence comprises decoding an event based on a value in a value register, when the context identifier is equal to the index, by generating an event in a first state if the value in the value register is less than the value assigned to the LPS range or by generating an event in a second state if the value in the value register is greater than or equal to the number.

17. (original) The method defined in Claim 16 further comprising performing renormalization in response to decoding the event only when the value in the value register is greater than or equal to the number.

18. (original) The method defined in Claim 12 wherein determining a value of a binary event based on the value of the range for the LPS and bits from an information sequence comprises decoding an event, when the context identifier is equal to the index, by first subtracting the value assigned to the LPS range from the range register and, where the event is generated in a first state if the value in the value register greater than or equal to a value in the range register or is generated in a second state if the value in the value register is less than the value in the range register.

19. (original) The method defined in Claim 18 further comprising performing renormalization in response to decoding the event only when the value in the value register is greater than or equal to the value in the range register.

20. (original) The method defined in Claim 18 wherein the value assigned to the range of the LPS is 2 if the context identifier is equal to the index.

21. (original) The method defined in Claim 19 wherein the value assigned to the range of the LPS is 100 Hex if the context identifier is equal to the index.

22. (original) The method defined in Claim 19 wherein the value assigned to the range of the LPS is 2 if the context identifier is equal to the index and wherein a last bit read in a bitstream containing binary events being decoded by the decoding engine is equal to 1.

23. (currently amended) An article of manufacture having one or more recordable media storing instructions thereon which, when executed by a system, cause the system to decode data by:

generating a context identifier for a binary event;

determining a value for a least probable symbol (LPS) ~~LPS~~ and a probability estimate for the LPS;

assigning a value to a range for the LPS, wherein the value is based on the probability estimate, a value stored in the range register and the context identifier to a range for the LPS if the context identifier is not equal to an index and the value is not based on the value stored in range register if the context identifier is equal to the index; and

determining a value of a binary event based on the value of the range for the LPS and bits from an information sequence.

24. (original) An arithmetic encoder comprising:

a probability estimator to generate a probability estimate that each event of an event sequence has a particular value, wherein the probability estimator generates the probability estimate in response to corresponding context information for said each event; and

a coding engine coupled to the probability estimator to generate zero or more bits of an information sequence in response to each event and its corresponding probability estimate, wherein the coding engine codes an event to signal the end of arithmetically encoded data in the information sequence using a constant for a subrange interval that is independent of a value of the range register prior to coding the end of slice signal.

25. (original) The encoder defined in Claim 24 wherein the coding engine using a constant to code the event to signal the end of the events in the event sequence enables inclusion of any remaining contents of a low register into the information sequence.

26. (original) The encoder defined in Claim 25 wherein the coding engine flushes any remaining contents of the low register and sets a last bit written during flushing equal to 1.

27. (original) A method for encoding data, the method comprising:

coding events in an event sequence to produce encoded data; and

generating a bitstream using the encoded data, including coding an indicator for use when decoding to indicate an end of arithmetically encoded data in the bitstream.

28. (original) The method defined in Claim 27 wherein non-arithmetically coded data follows the arithmetically encoded data in the bitstream.

29. (original) The method defined in Claim 27 wherein coding the indicator comprises coding an event to signal the end of the slice.

30. (original) The method defined in Claim 27 wherein coding the event to signal the end of slice comprises using a constant for a subrange interval that is independent of a value of the range register prior to coding the end of slice signal.

31. (original) The method defined in Claim 30 wherein coding the event to signal the end of a slice using a constant enables flushing of any remaining contents of a low register into the information sequence.

32. (original) The method defined in Claim 31 wherein flushing any remaining contents of the low registers comprises setting a last bit written during flushing equal to 1.

33. (original) An article of manufacture having one or more recordable media storing instructions thereon which, when executed by a system, cause the system to encode data by:

coding events in an event sequence to produce encoded data; and

generating a bitstream using the encoded data, including coding an event to signal an end of arithmetically encoded data in the bitstream using a constant for a subrange interval that is independent of a value of the range register prior to coding the end of slice signal.

34. (original) The article of manufacture defined in Claim 33 further comprising instructions which when executed by the system cause the system to flush contents of a low register, including setting a last bit written during flushing equal to 1.

35. (original) An apparatus for encoding data, the apparatus comprising:

means for coding a block of data to produce encoded data; and

means for generating a bitstream using the encoded data, including means for wherein the coding engine codes an event to signal an end of arithmetically encoded